Training Text

マイコントレーニングボード
MT-R300

キースキャン基板（MT-E503）編

VPort Lab. 代表
元長野県工科短期大学校教授
工学博士 星野　俊行　著

Sunhayato

安全上のご注意

このたびは弊社製品をご使用いただき、誠にありがとうございます。本項では、誤った取り扱いによる事故を未然に防ぐための安全上の注意事項を説明しています。弊社製品をご使用になる前に必ずお読みください。

⚠警告	この表記を無視して誤った取り扱いをすると、死亡や重傷など、人体への重大な障害をもたらす恐れのある内容について示しています。
⚠注意	この表記を無視して誤った取り扱いをすると、軽傷または中程度の障害をもたらす恐れのある内容について示しています。また、本製品や本製品に接続している機器に損傷を与える可能性がある事項についても示しています。

⚠警告

水場禁止	▶水分の多いところ、水がかかる場所では、本製品は使用しないでください 風呂場や台所など水分の多いところ、水がかかる場所では、本製品は使用しないでください。 火災、感電、故障の原因となります。
禁止	▶医療、軍事、航空宇宙、列車、運送、原子力などの制御設備へは使用しないでください 医療機器、軍事機器、航空宇宙機器、運送、原子力などの制御設備などの人命に関わるシステムへの使用は意図しておりません。
接触禁止	▶雷が鳴りはじめたらご使用をお控えください 近くに雷が発生したときは、パソコンから本製品を抜いてご使用をお控えください。また、ご使用のパソコンの取扱説明書に従って、電源プラグをコンセントから抜くなどしてください。雷によっては、火災、感電、故障の原因となることがあります。

⚠注意

プラグの差し込み	▶プラグは確実に差し込んでください 差し込みが不完全ですと火災、感電、過熱、故障の原因になります。
発火注意	▶発火、発煙、異臭への対処 発火、発煙、異臭がするなどの異常がありましたら使用を直ちに中止してください。そのまま使用すると、火災、故障の原因となります。 すぐにパソコンから抜き、煙などの異常が出なくなるのを確認し、販売店などに修理をご依頼ください。
分解禁止	▶分解・改造しないでください 分解、改造しないでください。怪我、感電、故障の原因となります。本製品の分解、改造による怪我や事故について、当社は責任を負いかねます。
接触禁止	▶濡れた手での操作は避けてください 濡れた手で電源ケーブル・プラグを抜き差ししないでください。また、製品に触れないでください。感電の原因となることがあります。

⚠注意

禁止	▶以下のような場所では使用しないでください 本製品を以下のような場所で使用すると、動作不良、故障の原因となります。 ・振動や衝撃が加わる場所 ・直射日光のあたる場所 ・湿気やホコリが多い場所 ・温度差の激しい場所 ・熱を発生するもの（暖房器具など）の近く ・強い磁力、電波が発生するもの（磁石、ディスプレイ、スピーカー、ラジオ、無線機など）の近く ・湿気の多い場所
子供注意	▶子供の手の届かない場所に置いてください 本製品に装着されている電子部品など子供が飲み込まないように注意してください。
引抜禁止	▶通信中はケーブルを引抜かないでください 本製品がパソコンと通信中の場合はケーブルを引抜かないでください。ケーブルの引抜きは、必ず通信していないときに行ってください。故障の原因になることがあります。
安全設計	▶安全設計をしてください 本製品を、高度な信頼性を必要とするシステムに使用する場合は、冗長設計、誤動作防止設計など充分な安全設計を必ず行ってください。本製品の故障、傷害により生じるいかなる損害、事故について当社は責任を負いかねます。
保管注意	▶長期間使用しない場合の保管について 長期間使用しない場合は、帯電防止袋などに入れ、ホコリなどが入らないようにしてください。ホコリや汚れが付着すると短絡、接触不良などの原因になります。
ホコリ注意	▶製品の清掃について 製品にホコリや汚れなどが付着すると短絡、故障の原因になりますので、下記の「▶お手入れについて」に従って清掃してください。
薬品注意	▶お手入れについて ホコリが付着した場合はサンハヤト製ジェットブロー（JBK-480）などのガススプレーで吹き飛ばしてください。
使用注意	▶故障、破損時の処理について 本製品が故障もしくは破損した場合は、速やかに使用を中止してください。そのまま使用しますと火災、感電、怪我の原因になるおそれがあります。
廃棄注意	▶本製品の廃棄について 本製品の廃棄は、各自治体の廃棄ルールに従ってください。詳しくは各自治体にお問い合わせください。

本資料についてのご注意

本資料について

・本資料は、電子工作や電子回路、パーソナルコンピュータの操作について一般的な知識をお持ちの方を対象にしています。
・本資料を元に操作するには、株式会社ルネサス エレクトロニクス製 H8/300H マイコンについての知識や開発環境などが必要です。
・Microsoft®、Windows® は米国 Microsoft 社の米国およびその他の国における登録商標です。
・その他、記載されている会社名、製品名は各社の商標または登録商標です。

本資料のご利用にあたって

・この取扱説明書に掲載している内容は、お客様が用途に応じた適切な製品をご購入頂くことを目的としています。その使用により当社及び第三者の知的財産権その他の権利に対する保証、又は実施権の許諾を意味するものではありません。また、権利の侵害に関して当社は責任を負いません。
・本資料に記載した情報を流用する場合は、お客様のシステム全体で充分評価し適用可能かご判断願います。当社では適用可能判断についての責任は負いません。
・本資料に記載してある内容は、一般的な電子機器（学習教材、事務機器、計測機器、パーソナル機器、コンピュータ機器など）に使用されることを目的としています。高い品質や信頼性が要求され、故障や誤作動が直接人命を脅かしたり人体に危害を及ぼす恐れのある、医療、軍事、航空宇宙、原子力制御、運輸、移動体、各種安全装置などの機器への使用は意図も保証もしておりません。
・この取扱説明書の一部、又は全部を著者および当社の承諾なしで、いかなる形でも転載又は複製されることは堅くお断りします。
・全ての情報は本資料発行時点のものであり、当社は予告なしに本資料に記載した内容を変更することがあります。
・この資料の内容は慎重に制作しておりますが、万一記述誤りによってお客様に損害が生じても当社はその責任を負いません。
・本資料に関してのお問合せ、その他お気付きの点がございましたら、当社までお問合せください。
・本資料に関する最新の情報はサンハヤト株式会社ホームページ（http://www.sunhayato.co.jp/）に掲載しております。

このテキストについて

本テキストでは、サンハヤト H8 マイコントレーニングボード MT-R300 のプログラム開発を、HEW（High-Performance Embedded Workshop）を使用して行った場合について説明しています。HEW の詳細な使用方法や注意事項につきましては、ルネサス エレクトロニクス社発行のマニュアルを参照してください。

サンハヤト H8 マイコントレーニングボード MT-R300 概要

- ターゲットマイコン　：H8/300H シリーズ 3062 グループ HD64H3062BF（以下 H8/3062BF マイコン）
- 書き込みボード　　：MT-R300（H8/3062BF マイコン モード 5（内蔵 ROM 有効拡張 16M バイトモード）、モード 7（シングルチップアドバンストモード）対応）

開発に必要なもの

- 開発用 PC（以下の条件を満たすもの）

PC 本体	Pentium Ⅲ 600MHz 以上を搭載した IBM PC/AT 互換機
OS	Windows XP, Windows Me, Windows 98SE, Windows 2000（Windows XP 推奨）
メモリ	128MB 以上（ロードモジュールの 2 倍以上）
ハードディスク	エミュレータ用ソフトウェアのインストールには 100MB 以上の空き容量が必要
入力デバイス	マウス、またはマウス相当のポインティングデバイス
インターフェイス	1 ポート以上の USB インターフェイス

- サンハヤト H8 マイコントレーニングボード　MT-R300
- 付属の USB ケーブル

このマニュアルで使用しているツール類

このスタートアップガイドでは、以下の OS、ツールのバージョンで動作したものとして説明しています。

- ホスト PC の OS　　：Windows XP Professional
- 統合化開発環境　　：HEW V.4.03.00001
- C コンパイラ　　：H8S,H8/300 Standard Toolchain (V.6.02 Release00)
- フラッシュ書き込みツール　：FDT（FlashDevelopmentToolKit ）Ver4.02

HEW のバージョンは、HEW の「 ヘルプ 」メニューの「High-Performance Embedded Workshop のバージョン情報 」で確認できます。

このマニュアルで紹介している各社サイト、ツールについて

本スタートアップガイドで紹介している各社サイトやツールは、本スタートアップガイド発行時のものを掲載しております。各社サイトの構成やツールのバージョン、またバージョンに伴ってツールのファイル名などが変わっている場合がありますのでご了承ください。

目　次

このテキストについて …… 5

サンハヤト H8 マイコントレーニングボード MT-R300 概要 …… 5
開発に必要なもの …… 5
このマニュアルで使用しているツール類 …… 5
このマニュアルで紹介している各社サイト、ツールについて …… 5

1. はじめに …… 7

2. 基礎知識 …… 8

2.1　キースキャン回路 …… 8

3. 接続図の概要 …… 9

4. 基本実習 …… 10

4.1　基本プログラム …… 10
実習 4.1.1　キー列をスキャンし、行列信号を取り込み LED に表示する …… 11
実習 4.1.2　キー列をスキャンし、行列信号を返す getgyoretu_key() 関数を作成する …… 12
実習 4.1.3　キー列をスキャンし、アスキーコード（押下しない時はヌル）を返す getch_key() 関数を作成する …… 12
実習 4.1.4　キー列をスキャンし、キーが押されるまで待って、アスキーコードを返す getchar_key() 関数を作成する …… 13
4.2　ライブラリの作成 …… 15
実習 4.2.1　キー列をスキャンし、キーが押されるまで待って、アスキーコードを取り込み、LED に表示する（ライブラリ作成） …… 15

5. おわりに …… 17

1. はじめに

キーボードは、各種制御機器の入力装置としてよく使われるため、そのハードウエアと制御プログラムを学習することは大切です。このボードは 16 個の押ボタンスイッチを用いて、4 行× 4 列のキー配列を構成しています。それをキースキャンという方法で制御する教育用ボードです。この方法は少ないビット数で多数のキーを制御できることが特徴です。

通常のキー配列は､キートップに文字が印刷済みのためそれを変更することは容易ではありません。このボードは、キー配列上の透明カバーに自作の文字盤を載せ替えることができるため、フレキシブルに必要な文字を対応させることができます。これもこのボードの大きな特徴です。例えば電卓のキーを模した配列にすることで、LCD ボードなどと組み合わせて電卓を構築することも可能です。

このボードを用いてキースキャン回路のハードウエアの概要を学習し、文字に対応する 16 進数のデータを取得し、ソフトウエアでそれをアスキーコードに変換するプログラムを作成します。

キー配列の縦方向と横方向に LED が設置されていますが、これは、キースキャン中、どの列のキーをスキャンし、どの行のキーが押されているかをモニターするためのものです。これは、プログラムのデバッグ時に大変便利なものです。これもこのボードの大きな特徴です。このボードを用いることでより効果的にキースキャン回路の学習ができます。

2. 基礎知識

2.1 キースキャン回路

直接 16 個のキーを入力する場合、外部ポートは 16 ビットも必要となり、あまり現実的ではありません。そこで少ないポートで入力できるように考えられたものがキースキャン回路です。これは、16 個のキーを 4 行× 4 列の配列状に設置し、ソフトウエアで各列を順次スキャンし、L レベルになる行とその列の交点から押されたキーを特定するものです。最後の列までスキャンしても見つからない場合はどのキーも押されていません。この方法では、外部ポートは行 4 ビットと列 4 ビットで合計 8 ビットになります。デコーダを用いてスキャン列を選択する場合には、列信号が 2 ビットとなり、合計 6 ビットになります。実際のキーの設置形状は必ずしも配列状でなくても構いません。

図 2-1 は、キースキャン基本回路を示します。全スイッチ（S1 ～ 16）が OFF のとき、行 0 ～ 3 の端子は、それぞれプルアップ抵抗により H レベルになります。列 2 をスキャンするときはそれを L レベル、他の列を H レベル、にします。そのとき、その列のスイッチ S7 を ON にすると、図のようにプルアップ抵抗経由で L レベルの列 2 側に電流が流れ、行 1 の端子だけが L レベルになります。同様に、スイッチ S3、S11、S15 を ON にするとそれぞれ対応する行 0、行 2、行 3 の端子だけが L レベルになります。スキャン列 2 以外のスイッチを ON にしても、プルアップ抵抗に電流が流れないため、行 0 ～ 3 の端子は H レベルのままです。従って、スキャン列と L レベルの行の交点のスイッチが ON と判断できます。誤って、例えば S7 と横並びのスイッチ（S8 など）を同時に ON にした場合、L レベルの列 2 と H レベルの他の列（列 3 など）がショート状態になります。これを防ぐために、各列にダイオードを挿入します。

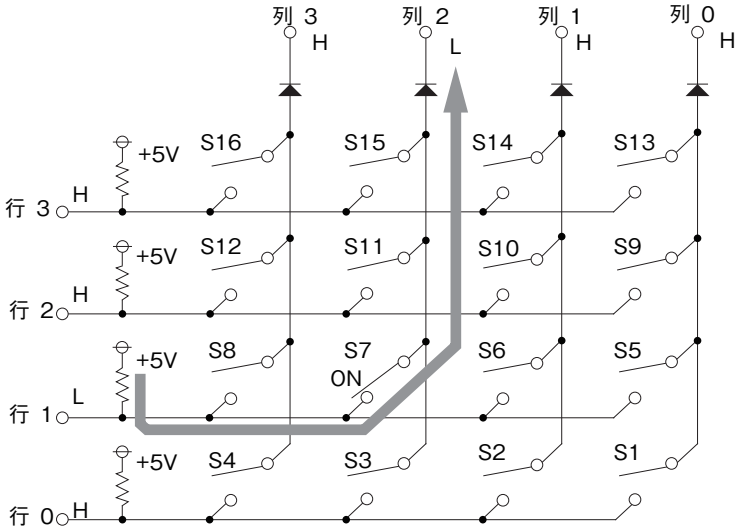

図 2-1　キースキャンの基本回路

3. 接続図の概要

図 3-1 は、このボードの接続図の概要を示します。ここでは、電卓用文字盤（http://www.sunhayato.co.jp/products/details.php?u=1357&id=02017）を使用した場合を示します。

VP1 の（b5､b4）は､スキャン列を選択する信号（以下「列信号」という）です。これを 2to4 デコーダ（74HC139）に入力し、その出力信号（負論理）をキー列 3 ～ 0 に供給し、スキャン列 1 つを選択します。スキャン列とキーの押下状況により、行のレベルを検知する信号（以下「行信号」という）が生成されます。これをシュミットトリガーインバーター（74HC14）で反転（波形整形）して、VP1 の（b3 ～ b0）に送ります。マイコンは列信号を制御し、列信号と行信号の合成信号（以下「行列信号」という）を取り込み押下キーを特定します。$\overline{OE}$ 信号（b6）は 0（許可）にしておきます。例えば、VP1 の列信号（b5、b4）が（10）（10 進数の 2）のとき、デコーダの B、A 入力端子にそれを入力し､出力端子 Y2（負論理）が L レベルになり､列 2 が選択されます。そのとき列 2 上のキー "2" を押すと、対応する行 1 のみが L レベルになります。これをシュミットトリガーインバーターで反転すると､VP1 の行信号（b3 ～ b0）は、（0010）になります。出力モードに設定された $\overline{OE}$ 信号（0）と列信号（10）は、マイコンでラッチされているため、VP1 の（b7 ～ b0）は行列信号（X0100010）となります。マイコンは、この信号から押下キー "2" を特定します。他のキーについても、対応する行列信号を考えてみましょう。列信号と行信号をモニターするために、それぞれモニター用 LED が設置されています。これによりデバッグがし易くなります。

接続図の詳細は MT-E503 取扱説明書（http://www.sunhayato.co.jp/products/item_data/MT-E503_Manual_SG087036.pdf）をご参照願います。

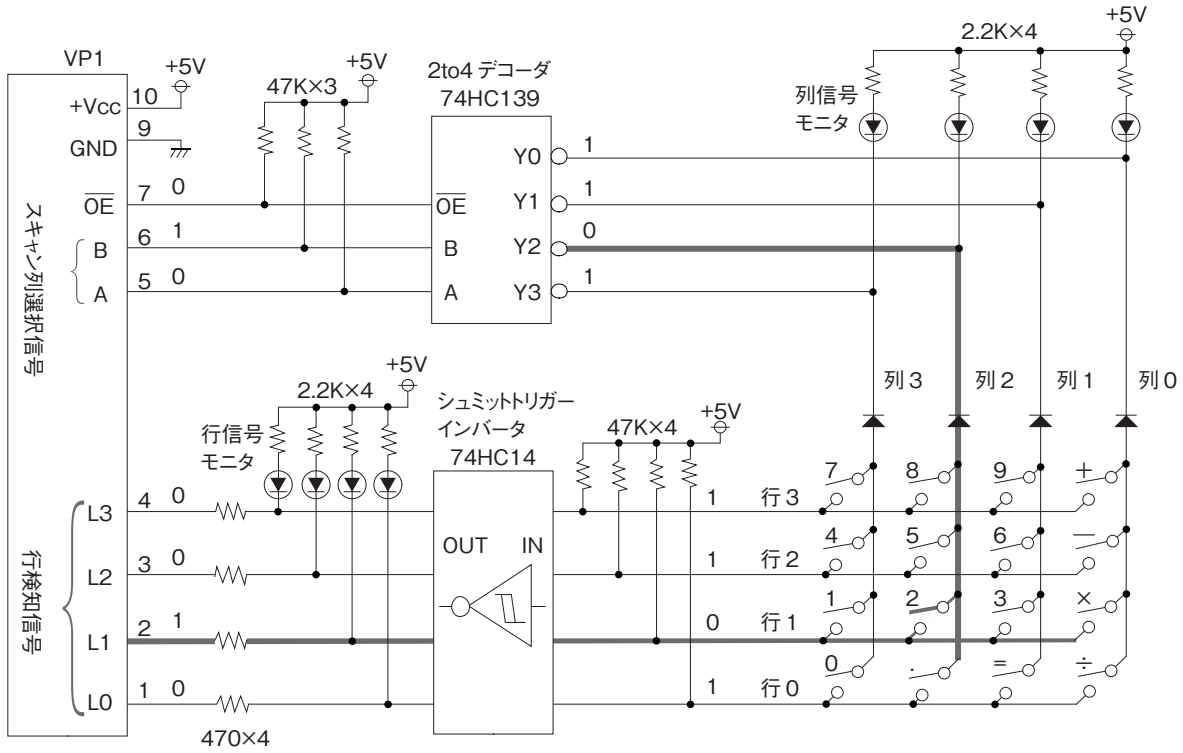

図 3-1　接続図の概要

4. 基本実習

ここでは、キー配列に電卓用文字盤を張り付けて実習します。まず基本プログラムを作成し、その後このボード用の関数（ライブラリー）を作成します。

4.1 基本プログラム

図 4-1 のように、付属の 10P ケーブルにより、マイコンボードの VPT とキースキャンボードの VP1 を接続し、キースキャンしその結果をマイコンに取り込みます。次にマイコンボードの VPE と標準 I/O ボードの VP2 を接続し、LED でその結果を確認します。

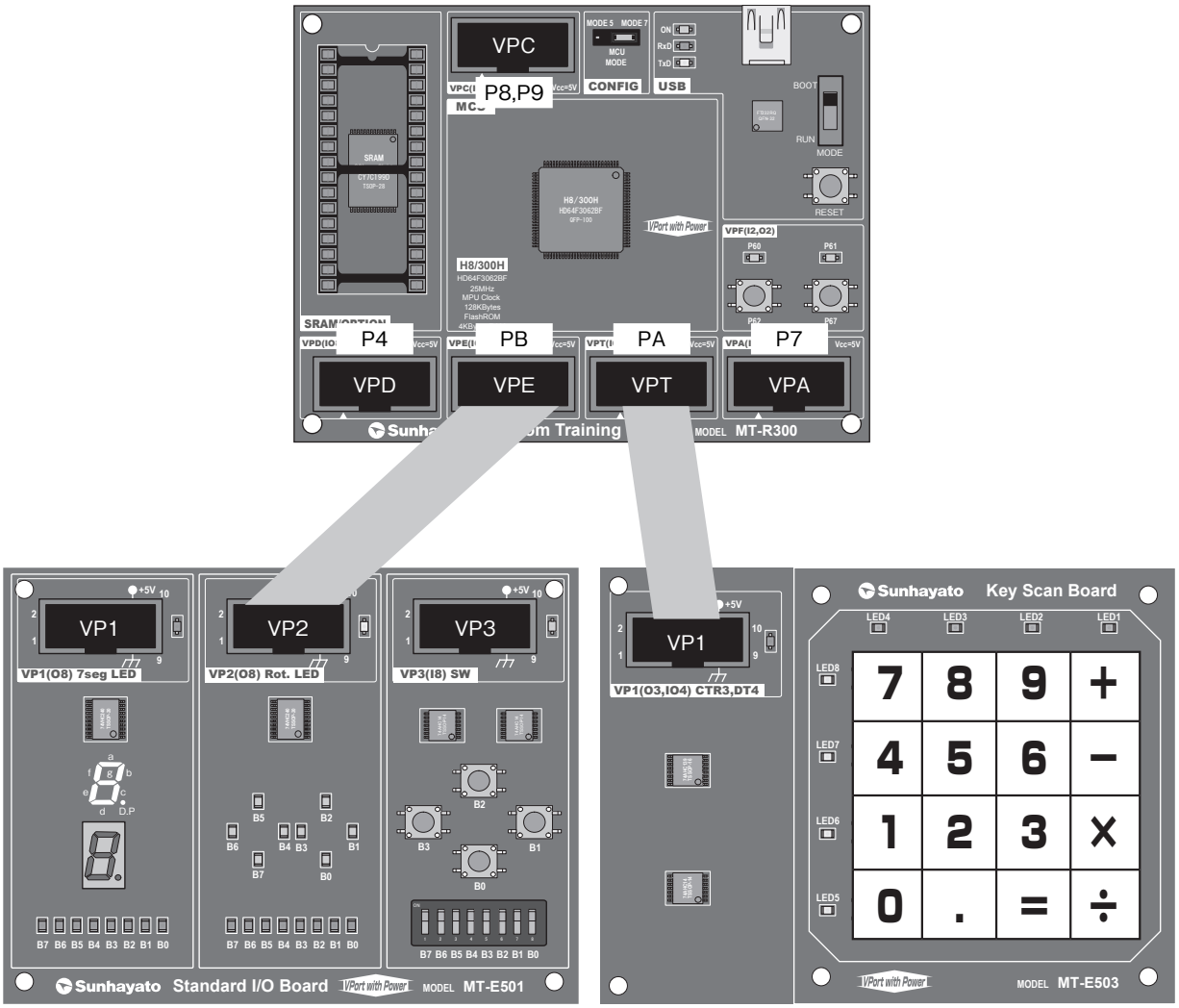

図 4-1　基板間の接続

ここでは、"X:¥xxxx¥workspace" にワークスペース、プロジェクト（key）を新規作成します。MT-E504 のテキストと同様にハードウエアのセットアップ関数とメイン関数は自動生成しません。手動で "main.c" ファイル（空白）を新規作成し、プロジェクトに追加します。既に作成済みの "X:¥xxxx¥workspace¥@library" フォルダー内の "mt_r300.c" もプロジェクトに追加します。

種々のヘッダーファイルをインクルードするために次のとおり相対パスを指定します。

(1) [ビルド]、[H8S、H8/300 Standerd TOOLchain] を選択し、"H8S、H8/300 Standerd TOOLchain" ウインドウに切り替えます。
(2) [標準ライブラリー] タグ、[カテゴリ] で " ソース "、[オプション項目] で " インクルードファイルディレクトリ "（デフォルト値）を確認する。
(3)［追加］ボタンを押下して、［相対パス］で "Project directory" を選択し、［OK］を押下する。
(4) 同様に [追加] ボタンを押下して、[相対パス] で "Costom directory" を選択し、ディレクトリーに相対パスで "..¥..¥..¥@library" と入力し、[OK] を押下する。

実習 4.1.1　キー列をスキャンし、行列信号を取り込み LED に表示する

[解説]

0 ～ 3 を 4 ビット左シフトした列信号（b5、b4）をデコーダーに入力し、その出力信号で、キー列を順次スキャンします。このときマイコンで、行信号（b3 ～ b0）を取り込むと、ラッチされている列信号（b5、b4）も同時に取り込まれます。この行列信号を LED に表示します。

[解答]

リスト 4.1.1 にプログラム例を示します。

リスト 4.1.1　キー列をスキャンし、行列信号を取り込み LED に表示する。

```
/** main.c  ********************************************************************************/
#include      <mt_r300.h>           // マイコンボード用ライブラリー
void main(void)
{
    char retu;                      // 列(3( 左端 ) ～ 0( 右端 )) 信号
    setup_VPT(0x70);          // キーの初期設定(b6: 出力(OE~),b5 ～ 4: 出力 ( 列 ),b3 ～ 0: 入力 ( 行 ))
    setup_VPE(0xff);          // 中央 LED の初期設定 ( 全ビット出力モード )
    while(1){
       out_VPT( retu << 4 );        // 0 ～ 3 をシフトした列信号 (b5,4) を出力する
       out_VPE( in_VPT() & 0x3f);   // 行列信号 (b5 ～ 0) を LED に表示する
       wait_ms(1000);                       // ゆっくり
       retu = (retu + 1) & 0x03;    // 次の列信号 (0 ～ 3) を計算し保存する
    }
}
```

キー列が順次スキャン（LED が順次点灯）することを確認します。ここで電卓用文字盤の "2" を押し続けます。列 2 がスキャンされたときだけ行 1 の LED が点灯することを確認します。このとき標準 I/O ボードの LED には、列信号（0x20）と行信号（0x02）を合成した行列信号（0x22）を表示します。同様に、他のキーを押したときの LED に表示される行列信号は次のようになります。キー文字 " × "、" ÷ " は、それぞれ "*"、"/" で代用しています。

表 4-1　キー文字、行列信号対応表

キー文字	/	*	-	+	=	3	6	9	.	2	5	8	0	1	4	7
行列信号	0x01	0x02	0x04	0x08	0x11	0x12	0x14	0x18	0x21	0x22	0x24	0x28	0x31	0x32	0x34	0x38

実習 4.1.2　キー列をスキャンし、行列信号を返す getgyoretu_key() 関数を作成する

[解説]

列 0 から 3 まで順次スキャンし、キーが押されている列で行列信号を返し、関数を終了します。3 列目までスキャンしてもキーが押されていない場合は、最後の行列信号 0x30 を返し、関数を終了します。関数を終了する前に、OE~ 信号を 1 にしてスキャンを禁止します（全列の LED が消灯）。

[解答]

リスト 4.1.2 にプログラム例を示します。

リスト 4.1.2　キー列をスキャンし、行列信号を返す getgyoretu_key() 関数を作成する。

```
/** main.c  ********************************************************************************/
#include      <mt_r300.h>
char getgyoretu_key(void)                   // 押下キーの行列信号（押下しないときは 0x30）を返す
{
    char retu, gyo;                         // 列信号，行信号（列信号も含む）
    retu = 0;                               // 列信号のクリア
    out_VPT(retu << 4);                     // 列信号の出力（1 件目）
    while(!((gyo=in_VPT() & 0x3f) & 0x0f) && ++retu<4){ // 行信号が 0 の間（列 3 まで）繰り返す
       out_VPT(retu << 4);                  // 列信号の出力（2 件目以降）
    }
    out_VPT(0x40);                          // スキャン禁止（b6:1）
    return gyo;
}
void main(void)
{
    setup_VPT(0x70);         // キーの初期設定（b6: 出力（OE~),b5 ～ 4: 出力（列），b3 ～ 0: 入力（行））
    setup_VPE(0xff);         // 中央 LED の初期設定（全ビット出力モード）
    while(1){
       out_VPE( getgyoretu_key() );// 押下キーの行列信号（押下しないときは 0x30）を表示
    }
}
```

実習 4.1.3　キー列をスキャンし、アスキーコード（押下しない時はヌル）を返す getch_key() 関数を作成する

[解説]

押下キーの行列信号から、対応するアスキーコードを検索する。16 個のアスキーコード配列を検索しても一致しなかった場合は、ヌルを返します。これは、押下されないか、複数キーを押下した場合です。

[解答]

リスト 4.1.3 にプログラム例を示します。

リスト 4.1.3　キー列をスキャンし、アスキーコード（押下しないときはヌル）を返す getch_key() 関数を作成する

```
/** main.c  ********************************************************************************/
```

```
#include      <mt_r300.h>
char getch_key(void) // キーのアスキーコード（押下されないときはヌル）を返す
{
// キー文字：'/', '*', '-', '+', '=', '3', '6', '9', '.', '2', '5', '8', '0', '1', '4', '7'
    char gyo_tb[]=    // 行列信号
       {0x01,0x02,0x04,0x08,0x11,0x12,0x14,0x18,0x21,0x22,0x24,0x28,0x31,0x32,0x34,0x38};
    char asc_code[] =// アスキーコード
       {0x2f,0x2a,0x2d,0x2b,0x3d,0x33,0x36,0x39,0x2e,0x32,0x35,0x38,0x30,0x31,0x34,0x37};
    int i;
    char retu, gyo, d;                              // 列信号，行信号（列信号も含む），データ
    retu = 0;                                       // 列信号のクリア
    out_VPT(retu << 4);                             // 列信号の出力（1 件目）
    while(!((gyo = in_VPT() & 0x3f) & 0x0f) && ++retu < 4){     // 行信号が 0 の間（列 3 まで）繰返す
       out_VPT(retu << 4);                          // 列信号の出力（2 件目以降）
    }
    out_VPT(0x40);                                  // スキャン禁止（b6:1）
    i = 0;
    while( gyo != gyo_tb[i] && ++i < 16 );          // 行列信号を検索する（一致しなかったら i=16）
    if(i < 16)
       d = asc_code[i];                             // 対応するアスキーコード
    else
       d = 0;                                       // 一致しなかたらヌル
    return d ;                                      // アスキーコードかヌルを返す
}
void main(void)
{
    setup_VPT(0x70);                // キーの初期設定（b6:出力（0E~),b5～4:出力（列),b3～0:入力（行））
    setup_VPE(0xff);                // 中央 LED の初期設定（全ビット出力モード）
    out_VPD(0);
    while(1){
       out_VPE( getch_key() );      // LED に押下キーのアスキーコードかヌルを表示する
       wait_ms(50);
       out_VPE(0);                  // 消灯する（高速でループしていることを確認するため）
       wait_ms(50);
    }
}
```

実習 4.1.4　キー列をスキャンし、キーが押されるまで待って、アスキーコードを返す getchar_key() 関数を作成する

[解説]

実習 4.1.3 の getch_key() 関数を活用して、キーが押されるまで待つ getchar_key() 関数を作成します。その概要は次のとおりです。1 つのキーを長く押しても 1 回の入力と判断します。既に処理済みのキー押下が長く続いている場合、キーから手を離してヌルになるまでは、取り込まれるデータを無視し続けます。ヌルの後、新たに取り込まれたデータを正しいものと判断し、それを返します。

[解答]

リスト 4.1.3 にプログラム例を示します。

リスト 4.1.3　キー列をスキャンし、キーが押されるまで待って、アスキーコードを返す getcha_key() 関数を作成する。

```
/** main.c  ********************************************************************************/
#include      <mt_r300.h>
char getch_key(void) // キーのアスキーコード（押下されないときはヌル）を返す
{
    char gyo_tb[]   = // 行列信号
       {0x01,0x02,0x04,0x08,0x11,0x12,0x14,0x18,0x21,0x22,0x24,0x28,0x31,0x32,0x34,0x38};
    char asc_code[] = // アスキーコード（文字で指定）
       { '/', '*', '-', '+', '=', '3', '6', '9', '.', '2', '5', '8', '0', '1', '4', '7'};
    int i;
    char retu, gyo, d;                                      // 列信号，行信号（列信号も含む），データ
    retu = 0;                                               // 列信号のクリア
    out_VPT(retu << 4);                                     // 列信号の出力（1 件目）
    while(!((gyo = in_VPT() & 0x3f) & 0x0f) && ++retu < 4){     // 行信号が 0 の間（列 3 まで）繰返す
       out_VPT(retu << 4);                                  // 列信号の出力（2 件目以降）
    }
    out_VPT(0x40);                          // スキャン禁止 (b6:1)
    i = 0;
    while( gyo != gyo_tb[i] && ++i < 16 ); // 行列信号を検索する（一致しなかったら i=16）
    if(i < 16)
       d = asc_code[i];                     // 対応するアスキーコード
    else
       d = 0;                               // 一致しなかたらヌル
    return d;                               // アスキーコードかヌルを返す
}
char getchar_key(void)                      // 新たにキーが押されるまで待ってアスキーコードを返す
{
    char d;
    while(getch_key());                     // 前のデータの連続入力を無視する（ヌルになるまで待つ）
    while(!(d = getch_key()));              // 新しいデータの入力（キーが押されるまで待つ）
    return d;                               // ヌルはない
}
void main(void)
{
    setup_VPT(0x70);         // キーの初期設定 (b6: 出力 (OE~),b5 ～ 4: 出力（列）,b3 ～ 0: 入力（行））
    setup_VPE(0xff);                        // 中央 LED の初期設定（全ビット出力モード）
    out_VPD(0);
    while(1){
       out_VPE( getchar_key() );            // 押下キーのアスキーコードを取り込み，LED でモニターする
       wait_ms(100);
       out_VPE(0);                          // 入力の区切りを示す
       wait_ms(100); }
```

```
}
```

前の実習では、アスキーコードのの配列を、asc_code[]={0x2f,0x2a,……}; のようにあえて 16 進数で設定しましたが、これは勉強のためです。ここでは、asc_code[]={ '/', '*',……}; のように簡単に文字で設定しました。どちらも同じことです。

1 つのキーを長く押しても、1 回しか取り込まないことを確認します。

4.2　ライブラリの作成

4.1 と同様にマイコンボードとキースキャンボード、標準 I/O ボードを接続します。"X:¥xxxx¥workspace" フォルダーにワークスペース、プロジェクト "key_lib" を新規作成します。

4.1 の実習で作成した関数をまとめたライブラリー "mt_e503.c", "mt_e503.h" を新規作成します。これらは、"mt_r300.c","mt_r300.h" などと同じフォルダー "X:¥xxxx¥workspace¥@library" に配置します。

ここでは、"mt_r300.c" と "mt_e503.c" をプロジェクトに追加します。

実習 4.2.1　キー列をスキャンし、キーが押されるまで待って、アスキーコードを取り込み、LED に表示する（ライブラリ作成）

[解答]

リスト 4.3.1a にプログラム例を示します。

リスト 4.3.1　キー列をスキャンし、キーが押されるまで待って、アスキーコードを取り込み、LED に表示する（ライブラリ作成）

```
/** main.c  ********************************************************************************/
#include      <mt_r300.h>          // マイコンボード用ライブラリー
#include      <mt_e503.h>          // キースキャンボード用ライブラリー
void main(void)
{
    setup_VPT(0x70);               // キーの初期設定(b6:出力(OE~),b5～4:出力(列),b3～0:入力(行))
    setup_VPE(0xff);               // 中央 LED の初期設定 ( 全ビット出力モード )
    while(1){
       out_VPE( getchar_key() );   // アスキーコードを取り込み, LED でモニターする
       wait_ms(100);
       out_VPE(0);                 // 入力の区切りを示す
       wait_ms(100);
    }
}
```

```
/** mt_e503.c ******************************************************************************/
#include      <iodefine.h>
#include      <mt_r300.h>          // マイコンボード用ライブラリー
#include      <mt_e504.h>          // 7 セグメント LED ボード用ライブラリー
/*****************************************************************************************/
/*  キースキャンボードの初期設定関数（VPT のセットアップのみ）                              */
/*****************************************************************************************/
void initialize_key(void)
```

```
{
    setup_VPT(0x70);                // キーの初期設定(b6:出力(0E~),b5～4:出力(列),b3～0:入力(行))
}
/*****************************************************************************************/
/*  キースキャンし，押下キーのアスキーコード(押下しない時はヌル)を返す関数                  */
/*****************************************************************************************/
char getch_key(void)                // 押下キーのアスキーコード(押下しない時はヌル)を返す関数
{
    char gyo_tb[]    =              // 行列信号
        {0x01,0x02,0x04,0x08,0x11,0x12,0x14,0x18,0x21,0x22,0x24,0x28,0x31,0x32,0x34,0x38};
    char asc_code[] =               // アスキーコード
        { '/', '*', '-', '+', '=', '3', '6', '9', '.', '2', '5', '8', '0', '1', '4', '7'};
    int i;
    char retu, gyo, d;              // 列信号，行信号(列信号も含む)，データ
    retu = 0;                       // 列信号のクリア
    out_VPT(retu << 4);             // 列信号の出力（1件目）
    while(!((gyo = in_VPT() & 0x3f) & 0x0f) && ++retu < 4){     // 行信号が0の間(列3まで)繰返す
       out_VPT(retu << 4);                  // 列信号の出力（2件目以降）
    }
    out_VPT(0x40);                          // スキャン禁止(b6:1)
    i = 0;
    while( gyo != gyo_tb[i] && ++i < 16 ); // 行列信号を検索する(一致しなかったらi=16)
    if(i < 16)
       d = asc_code[i];                     // 対応するアスキーコード
    else
       d = 0;                               // 一致しなかたらヌル
    return d;                               // アスキーコードかヌルを返す
}

/*****************************************************************************************/
/*  新たにキーが押されるまで待ってアスキーコードを返す                                    */
/*****************************************************************************************/
char getchar_key(void)
{
    char d;
    while( getch_key() );               // 前のデータの連続入力を無視する(ヌルになるまで待つ)
    while(!(d = getch_key()));          // 新しいデータの入力(キーが押されるまで待つ)
    return d;                           // ヌルはない
}
```

```
/** mt_e503.h ************************************************************************/
void initialize_key(void);
char getch_key(void);
char getchar_key(void);
```

5. おわりに

本ボードは、列と行のモニター LED によりキースキャンの動作を確認できるため、キースキャン動作やその制御プログラムが理解し易かったと思います。ここで作成した char getchar_key(void) 関数は、返値がアスキーコードのためネイティブな C 言語と同じ感覚で使えます。MT-E502 の LCD ボードなどと組み合わせて、キーボードの文字を表示することも簡単にできるため、電卓などに発展できます。これらの応用的な実習は別のテキストに譲ります。

マイコントレーニング基板　MT-R300
トレーニングテキスト　キースキャン基板編

発行日　　2010-10-10　Rev1.00
発　行　　サンハヤト株式会社
〒 170-0005　東京都豊島区南大塚 3 丁目 40 番 1 号

©2010 Sunhayato Corp. All rights reserved.

SG107015